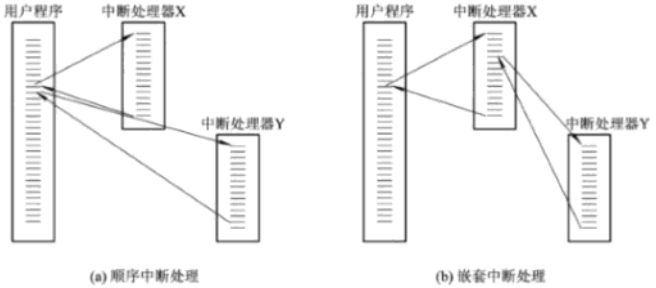
6中断、驱动、软件

2018年12月17日　　10:45

-
- 中断机构和中断处理程序

一. 中断简介

1. 中断和陷入
   1) 中断/外中断：CPU对外部IO设备发来的中断信号的响应
   2) 陷入trap/内中断：CPU内部时间引起的中断，如运算溢出、地址越界、电源故障
2. 中断向量表和中断优先级
   1) 中断向量表：为每种设备匹配以相应的中断处理程序，把其入口放在一个表项，为每个中断请求规定一个中断号，对应一个表项。由中断控制器根据中断号查找中断向量表，再转入中断处理程序
   2) 中断优先级：根据服务紧急度给不同中断信号源排级，如键盘<打印机<磁盘
3. 对多中断源的处理方式
   1) 屏蔽/禁止中断：处理一个中断时让所有新到中断请求都等待
   2) 嵌套中断：优先响应高优先级的中断请求，高级的可以抢占低的处理机
   3)



二. 中断处理程序

1. 请求IO操作的进程会被挂起，IO完成后，设备控制器会向cpu发送中断请求
2. 中断处理步骤
   1) 测定是否有未响应的中断信号。每当设备完成字符操作，设备控制器便发出中断请求，将数据在设备和内存缓冲之间交换。因此cpu每执行完一条指令都要测定是否有中断
   2) 保护被中断进程的cpu环境。把处理机状态字psw和程序计数器pc的下条指令的地址存到中断保留栈，把cpu寄存器内容压入中断栈

(1)

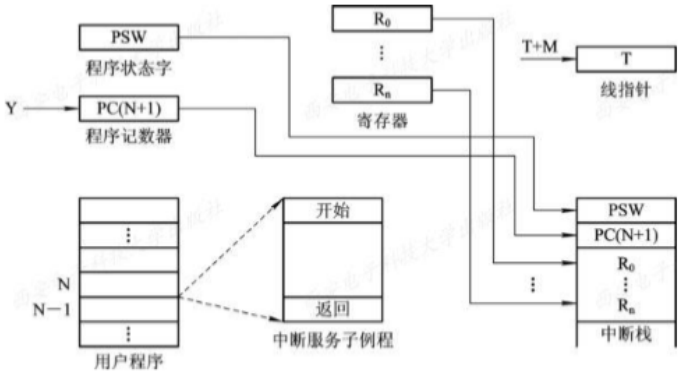


图6-10 中断现场保护示意图

3) 转入相应的设备处理程序。cpu确认中断信号源，向它发送确认信号，让设备取消中断请求信号，之后把中断程序地址装入程序计数器
4) 中断处理。cpu从程序计数器找到中断程序，从设备控制器读设备状态，判断正常完成中断/异常结束中断，在内存和设备间交换数据/处理异常
5) 恢复cpu线程并退出中断
   (1) 若是屏蔽中断方式，就回到被中断的进程，从中断栈读出现场
   (2) 若是嵌套中断，会处理优先级更高的中断请求
6)

图6-11 中断处理流程

3. 除了4）外都是相同的，所以unix把1235集中起来，形成中断总控程序

-
- 设备驱动程序

一. 设备驱动程序概述

1. 设备驱动程序的功能
   1) 接受设备无关软件发来的命令、参数，转换成设备相关操作序列
   2) 检查请求合法性、了解设备工作状态、传递参数、设置设备工作方式
   3) 发出IO命令，根据设备空闲与否，立即启动或挂在设备队列
   4) 及时响应设备控制器的中断请求，调用对应中断处理程序
2. 设备驱动程序的特点
   1) 是低级的系统例程
   2) 是负责在设备无关软件和设备控制器之间通信和转换的程序
   3) 与设备控制器及IO硬件特效紧密相关，不同设备应有不同驱动
   4) 与IO设备的IO控制方式紧密相关，主要有中断驱动和DMA两种
   5) 一部分必须有汇编写，以做到与硬件紧密相关，有的还固化在ROM中
   6) 可重入，可能会在一次调用完成前再次被调用
3. 设备处理方式
   1) 为每类设备设一个进程，执行其IO，适用于大系统，如交互终端，打印机各一个
   2) 整个系统只有一个IO进程，或IO各一个
   3) 不专门设置设备处理进程，直接给用户或系统调用驱动，目前较常用

二. 设备驱动程序的处理过程

1. 将抽象要求转换为具体命令、参数。如盘块号转换成盘面、磁道号、扇区，只有驱动能做到
2. 校验服务请求是否合法，并终止进程或通知有错。如从打印机读数据，修改只读文件

3. 检查设备状态，指启动设备前，测试状态寄存器各状态，可能使进程等待

1)

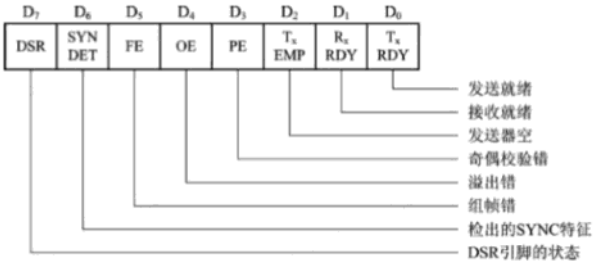


图6-12 状态寄存器中的格式

4. 向各寄存器传送命令、参数。如控制命令、传送速率、字符长度
5. 启动IO设备，向控制器的命令寄存器传控制命令
    1) 多道程序中，驱动程序一旦发出IO命令，启动操作后，驱动就把控制返回给IO系统，阻塞自己，到下个中断来时再被唤醒，IO操作是在设备控制器的控制下进行的，此时处理机可以并行干其他事

三. 对IO设备的控制方式

1. **轮询的可编程IO**方式：cpu向控制器发出IO指令时，把状态寄存器的忙闲标志busy置1，之后不断循环测试busy，是0时表示操作完成了
    1) 绝大部分时间在轮询测试，浪费cpu
2. **使用中断的可编程IO**方式：cpu发出IO命令后立即继续执行其他任务，由控制器完成指令后用控制线发送中断信号，cpu确认是正常完成中断后再处理
    1) cpu与io并行，利用率一般成百倍以上的提升

2)

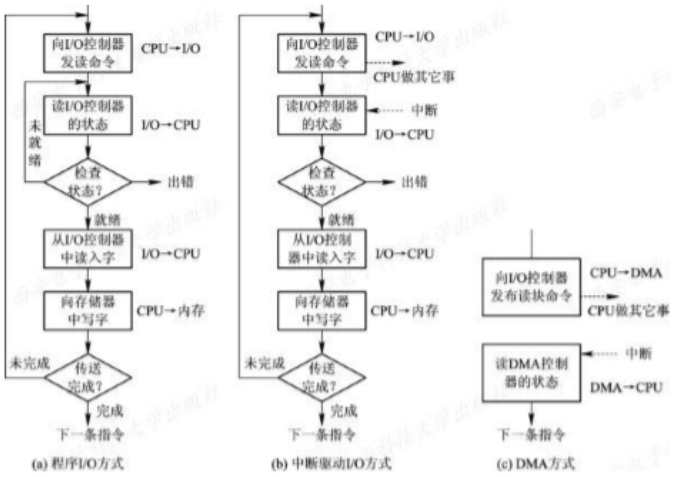


图6-13 程序I/O和中断驱动方式的流程

3. **直接存储器访问**方式(Direct Memory Access)
    1) 直接存储器访问方式的引入
        (1) 数据传输基本单位是数据块，每次至少传一个数据块
        (2) 传输数据时直接在**设备和内存间交换**的
        (3) 仅在传送数据块的开始和结束时才需cpu干预，由DMA控制传输
    2) DMA控制器的组成：与主机的接口、与块设备的接口、IO逻辑

(1)

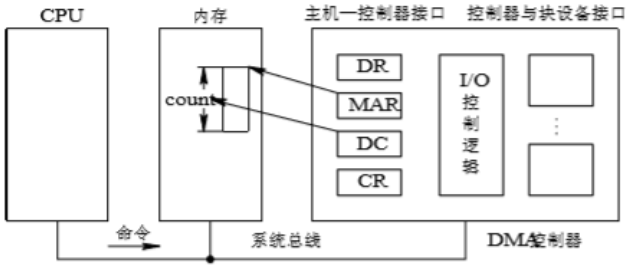


图 5-8 DMA 控制器的组成

(2) 命令/状态寄存器CR：cpu发来的IO命令、控制信息、设备状态

(3) 内存地址寄存器MAR：输入的内存起址，或输出的设备源址

(4) 数据寄存器DR：暂存待交换的数据

(5) 数据计数器DC：存放要操作的字数

3) DMA工作过程

(1) cpu向磁盘控制器发送命令，命令存进CR，地址存进MAR，字数存进DC，磁盘源址送进IO逻辑，启动DMA，数据送入DR，传到MAR指向的单元，--DC的内容，直至为0后发出中断请求

(2)

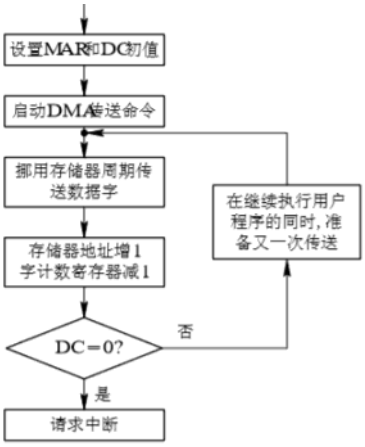


图 5-9 DMA 方式的工作流程图

4. **IO通道控制**方式

1) 通道方式的引入：DMA的发展，读写单位由一个数据块变成一组数据块

2) 通道程序：由一系列通道指令/命令构成，每条都包含：操作码（读/写/控制）、内存首地址、字数、通道程序结束位P（是不是通道最后一条指令）、记录结束标志（是不是一段指令的最后一条）

(1)

操　作	P	R	计　数	内存地址
WRITE	0	0	80	813
WRITE	0	0	140	1034
WRITE	0	1	60	5830
WRITE	0	1	300	2000
WRITE	0	0	250	1650
WRITE	1	1	250	2720

◆

◆ 与设备无关的IO软件

一. 与设备无关软件的基本概念(Device Independence)

1) 设备独立性/设备无关性：应用程序独立于具体使用的物理设备

2) 为提高 OS 的可适应性和可扩展性，在现代 OS 中都毫无例外地实现了它

1. 以物理设备名使用设备：早期OS必须用物理名称来控制设备

1）不灵活，不利于提高IO设备利用率

2. 引入逻辑设备名，实现IO重定向：不必更改程序，也可更换设备

3. 逻辑设备名到物理设备名的转换：程序执行时，需把逻辑地址转换为物理地址

二. 与设备无关的IO软件

1. 设备驱动程序的同一接口：方便添加新设备驱动程序，将抽象设备名映射到合适的驱动程序，保护设备，防止无权用户的访问

2. 缓冲管理：配置字符设备、块设备的缓冲

3. 差错控制：暂时性错误如电源波动等，用重试操作来纠正；持久错误如掉电，磁盘划痕、除以零等。磁盘盘块遭破坏只需记录其块号，以后不再使用即可

4. 对独立设备的分配与回收

   1）确认空闲再分配，否则阻塞它，等被释放了再唤醒

5. 独立于设备的逻辑数据块：隐藏不同设备的不同数据交换单位，向高层软件提供统一大小的逻辑数据块

三. 设备分配

1. 设备分配中的数据结构

   1）设备控制表DCT

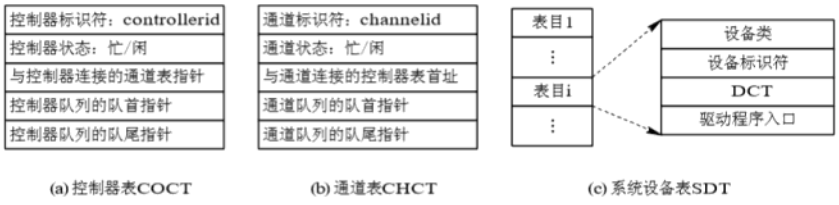
   (1)

图 5-20 设备控制表

   (2) 表项包含：设备类型、设备标识、忙闲状态、控制器表指针、重复执行次数（达到指定大小后认为传送失败）、请求PCB队列指针

   2）控制器控制表(COCT)、通道控制表(CHCT)和系统设备表(SDT)

   (1)



(a) 控制器表COCT (b) 通道表CHCT (c) 系统设备表SDT

图 5-21 COCT、CHCT 和 SDT

2. 设备分配时应考虑的因素

   1）设备的固有属性：独占设备：直至进程释放前都由当前进程独占、共享设备：对访问先后次序合理调度、虚拟设备：同共享设备

   2）设备分配算法：先来先服务、高优先级优先

   3）设备分配中的安全性

   (1) 安全分配方式：发出 I/O 请求后，便进入阻塞状态，直到其 I/O 操作完成时才被唤醒。摒弃了造成死锁的的“请求和保持”条件。缺点是进程进展缓慢，即 CPU 与 I/O 设备是串行工作的

   (2) 不安全分配方式：发出 I/O 请求后仍继续运行，需要时又发出新请求，仅当进程所请求的设备已被另一进程占用时，才进入阻塞状态。 这种分配方式的优点是，一个进程可同时操作多个设备，推进迅速。缺点是可能造成死锁。因此，还应进行安全性计算

3. 独占设备的分配程序
    1) 基本的设备分配程序
        (1) 分配设备：根据 I/O 请求中的物理设备名，查找系统设备表(SDT)，从中找出其 DCT，若 DCT 中的设备状态字段为忙，或不通过安全性计算，便将 PCB 挂在设备队列上；安全才将设备分配给请求进程
        (2) 分配控制器：把设备分配给请求 I/O 的进程后，再到其 DCT 中找出与该设备连接的控制器的 COCT，根据忙闲字段，将 PCB 挂在该控制器的等待队列上或将该控制器分配给进程
        (3) 分配通道：在 COCT 中又可找到与该控制器连接的通道的 CHCT，再根据 CHCT 的忙闲字段，将 PCB 挂在该通道的等待队列上，或将该通道分配给进程
        (4) 只有在设备、 控制器和通道三者都分配成功时，这次的设备分配才算成功。 然后，便可启动该 I/O 设备进行数据传送
    2) 设备分配程序的改进
        i. 上述程序不具备设备无关性：以物理设备名来提出 I/O 请求；采用单通路的 I/O 系统结构，容易产生“瓶颈”现象。为此，应从以下两方面改进，提高灵活性和分配的成功率
        (1) 增加设备的独立性：用逻辑设备名请求 I/O。系统先从 SDT 中找出空闲的该类设备的 DCT，仅当所有该类设备都忙时，才把进程挂在其的等待队列上
        (2) 考虑多通路情况：仅当所有的控制器(通道)都忙时，此次的控制器(通道)分配才算失败，才把进程挂在控制器(通道)的等待队列上

四. 逻辑设备名到物理设备名映射的实现
1. 逻辑设备表(Logical Unit Table)
    1) 每个表目中包含三项： 逻辑设备名、物理设备名和设备驱动程序的地址
    2) 当系统为进程分配设备时，将在 LUT 上建立一个表目
    3) 以后再有进程请求该逻辑设备名时，系统便可查找 LUT，找到物理设备
2. LUT的设置问题
    1) 只设一张：不允许重复逻辑设备名，只适合单用户系统，左图
    2) 每个用户设一张：每个PCB设一张，右图
    3)

逻辑设备名	物理设备名	驱动程序入口地址
/dev/tty	3	1024
/dev/printe	5	2046
⋮	⋮	⋮

(a)

逻辑设备名	系统设备表指针
/dev/tty	3
/dev/printe	5
⋮	

(b)

图 5-19　逻辑设备表

i.
ii.
iii.
iv.
v.

vi.
vii.
viii.
ix.
x.
xi. ---------我是底线-------